# NodeMCU

After flashing NodeMCU firmware it is recommended to erase filesystem:

```
file.format()
```

This firmware gives LUA command line on serial console. There is no direct way to upload files with lua script into module. Instead of this, file is created using lua command: open file, write some strings, close file.

```
file.open("init.lua","w")
file.writeline([[print("WIFI control")]])
file.writeline([[wifi.setmode(wifi.SOFTAP)]])
file.close()
```

This method is not comfortable, better is to use tool like http://esp8266.ru/esplorer/ (source on github) or https://github.com/GeoNomad/LuaLoader python script

Online page to make online custom builds: custom builds: http://frightanic.com/nodemcu-custom-build/

- https://github.com/nodemcu/nodemcu-firmware/wiki
- https://github.com/nodemcu/nodemcu-flasher
- https://github.com/nodemcu/nodemcu-firmware/releases

# Programming

- Devel API (latest): https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en-%28dev096%29
- GPIO MAP: https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en-%28dev096%29#new_gpio_map
- Current API: https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en

## Saving memory

- Avoid upvalues for context passed between event callbacks, as its very difficulat to get a handle on memory leaks created by these. Only use globals for this usecase.
- Nil globals once they are no longer needed so that they can be properly GCed.
- Allocate resources and create closures on a just-in-time basis.
- The cost of require or dofile is relatively small, so break your program into overlays one per event and use a small stub function as a callback to load each.
    - Use compiled .lc files and load using dofile().

## Scan WiFi

```
function listap(t)
  for k,v in pairs(t) do
    print(k.." : "..v)
  end
end
wifi.sta.getap(listap)
```

## Connect to WiFi

```
wifi.setmode(wifi.STATION)
wifi.sta.config("HOMENET","mysecret")
print(wifi.sta.getip())
```

## Scan 1 wire bus

```
ow.setup(1)
print (ow.reset(1)) -- print 1 if device found
```

## Resolving DNS

It is not so easy. It need socket instance which will be used only for one DNS request, then socket will be destroyed. Response is performed by callback. There is sth about easier API: https://github.com/nodemcu/nodemcu-firmware/issues/189

## Compile to byte code

You can compile everytihn except init.lua.

```
node.compile("myprog.lua")
```

Will produce myprog.lc.

# Other Projects

- 1-wire temp sensor reporting to http://thingspeak.com, http://hacklab.fi/news/esp8266-ds18b20-thingspeak-nodemcu/
- 1-wire temp sensor reporting to http://nettemp.pl, http://techfreak.pl/bezprzewodowe-czujniki-na-esp8266-nettemp/

- http://techfreak.pl/bezprzewodowe-czujniki-temperatury-ds18b20-na-esp8266/
- 1-wire temp sensor https://blog.jokielowie.com/en/2015/10/domoticz-cz-2-termometr-wifi-z-precyzja-do-dwoch-miejsc-po-przecinku-czyli-esp8266-dla-poczatkujach-w-praktyce/
- Control 2 GPIO from Web http://randomnerdtutorials.com/esp8266-web-server/
- Http server https://github.com/marcoskirsch/nodemcu-httpserver
- Remote update http://www.instructables.com/id/ESP8266-WiFi-File-Management/ https://github.com/breagan/ESP8266_WiFi_File_Manager

# Free Cloud/Server services

- Thingspeak: http://thingspeak.com/
- Telit: http://www.telit.com/products-and-services/iot-platforms/iot-portal
- Google: https://cloud.google.com/solutions/iot/
- IBM: https://internetofthings.ibmcloud.com/#/

todo

- https://www.google.com/search?q=ai+cloud+esp8266&ie=utf-8&oe=utf-8